Exhibit B

NV3 Graphics Reference Manual

Don Bittel
Kevin Dawallu
Chris Malachowsky

NVidia Corporation

1226 Tiros Way

Sunnyvale, CA 94086

Version $Revision:

$Date:

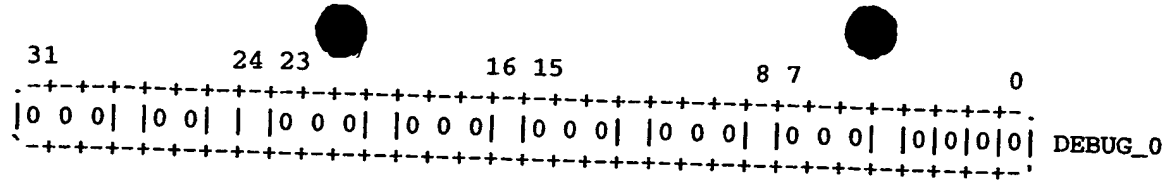
/home/nv3/CVS/manuals/$RCSfile: dev_graphics.ref,v $

4 - DEBUG REGISTERS

The DEBUG registers are used to reconfigure NV during debug or chip testing. These registers may contain function disable bits and hidden context. Only the BIOS should access these registers. In essence these are CYA bits.

DEBUG_0 register should only be written when PGRAPH:STATUS:STATE is Idle. Waiting for idle is really only important if one of the optimization enables is going to be changed and rendering activity is in progress. DEBUG_0 can be read at any time.

The CACHE_STATE bit will reset only the texture cache state machines in the graphics engine. This should not affect any context state. Once set to Reset, it will clear itself after the reset operation is complete.

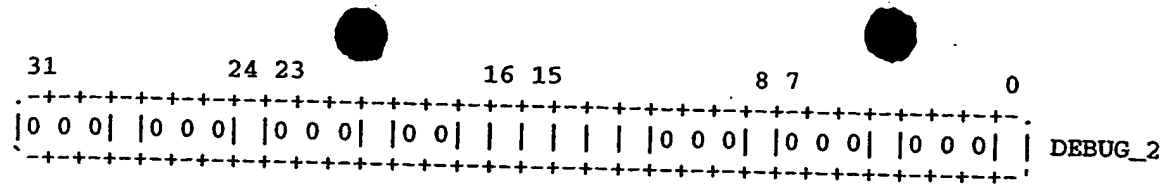



```
#define NV_PGRAPH_DEBUG_0                                   0x00400080 /* RW-4R */
```

```
#define NV_PGRAPH_DEBUG_0_CACHE_STATE                             2:2 /* CW-VF */
#define NV_PGRAPH_DEBUG_0_CACHE_STATE_NORMAL                0x00000000 /* CW--V */
#define NV_PGRAPH_DEBUG_0_CACHE_STATE_RESET                 0x00000001 /* -W--V */
```

DEBUG_2 register should only be written when PGRAPH:STATUS:STATE is Idle. Waiting for idle is really only important if one of the optimization enables is going to be changed and rendering activity is in progress. DEBUG_2 can be read at any time.

The PREFETCH enable bit enables texture prefetching in the D3D0/D3D0Z classes when enabled.




```
#define NV_PGRAPH_DEBUG_2                                   0x00400088 /* RW-4R */
```

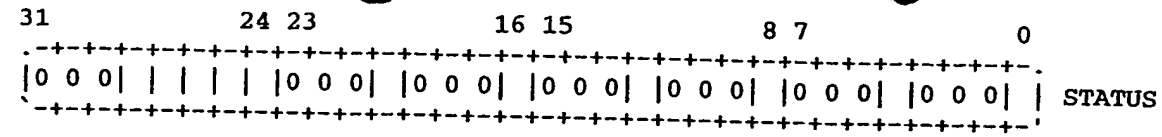
```
#define NV_PGRAPH_DEBUG_2_PREFETCH                             24:24 /* RWIVF */
#define NV_PGRAPH_DEBUG_2_PREFETCH_DISABLED                0x00000000 /* RWI-V */
#define NV_PGRAPH_DEBUG_2_PREFETCH_ENABLED                 0x00000001 /* RW--V */
```

The STATUS register provides access to internal graphics engine status information. This register is a read-only and can be read and written at any time.

The PORT_DMA bit indicates that the user device port is Busy awaiting the graphics engine to complete a dma operation that is in progress.

The DMA_ENGINE bit indicates that the integral graphic DMA engine is busy with a dma operation.

The DMA_NOTIFY bit signifies that a notification has been scheduled to be sent to the DMA engine. If DMA_NOTIFY is set to Busy then the DMA engine has not yet accepted the request. Any write to PGRAPH:NOTIFY will clear the scheduled notification.

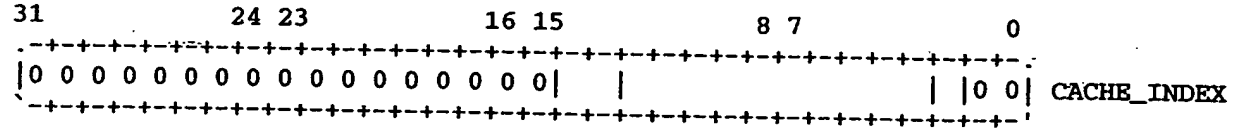



```
#define NV_PGRAPH_STATUS                                        0x004006B0 /* R--4R */
```

```
#define NV_PGRAPH_STATUS_PORT_DMA                                    16:16 /* R-IVF */
#define NV_PGRAPH_STATUS_PORT_DMA_IDLE                         0x00000000 /* R-I-V */
#define NV_PGRAPH_STATUS_PORT_DMA_BUSY                         0x00000001 /* R---V */
#define NV_PGRAPH_STATUS_DMA_ENGINE                                  17:17 /* R-IVF */
#define NV_PGRAPH_STATUS_DMA_ENGINE_IDLE                       0x00000000 /* R-I-V */
#define NV_PGRAPH_STATUS_DMA_ENGINE_BUSY                       0x00000001 /* R---V */
#define NV_PGRAPH_STATUS_DMA_NOTIFY                                  20:20 /* R-IVF */
#define NV_PGRAPH_STATUS_DMA_NOTIFY_IDLE                       0x00000000 /* R-I-V */
#define NV_PGRAPH_STATUS_DMA_NOTIFY_BUSY                       0x00000001 /* R---V */
```

The CACHE_INDEX register is provided to facilitate the writing and reading of the large rams used in the graphic engine's texture cache logic. These rams store 16-bit texture data. In order to access the CACHE_RAM register, the CACHE_INDEX register must first be written.
Two of the four banks are accessed at any time by a 32 bit accesses.

This register should only be accessed when PGRAPH:STATUS:STATE is Idle regardless or whether it is a erad or a write access. This register does not need to be context switched. It is provided for diagnostic accesses only.

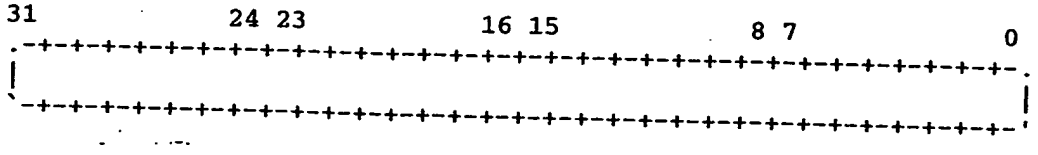



```
#define NV_PGRAPH_CACHE_INDEX                          0x004006c0 /* RW-4R */
#define NV_PGRAPH_CACHE_INDEX_BANK                            2:2 /* RWXVF */
#define NV_PGRAPH_CACHE_INDEX_BANK_10                  0x00000000 /* RW--V */
#define NV_PGRAPH_CACHE_INDEX_BANK_32                  0x00000001 /* RW--V */
#define NV_PGRAPH_CACHE_INDEX_ADRS                           12:3 /* RWXVF */
#define NV_PGRAPH_CACHE_INDEX_ADRS_0                   0x00000000 /* RW--V */
#define NV_PGRAPH_CACHE_INDEX_ADRS_1024                0x00000400 /* RW--V */
#define NV_PGRAPH_CACHE_INDEX_OP                            14:13 /* RWXVF */
#define NV_PGRAPH_CACHE_INDEX_OP_WR_CACHE              0x00000000 /* RW--V */
#define NV_PGRAPH_CACHE_INDEX_OP_RD_CACHE              0x00000001 /* RW--V */
#define NV_PGRAPH_CACHE_INDEX_OP_RD_INDEX              0x00000002 /* RW--V */
```

Figure 9-18 Cache Ram Index register

The CACHE_RAM register is provided to facilitate access to the data access port for the writing and reading of the large rams used in the graphic engine's texture cache.

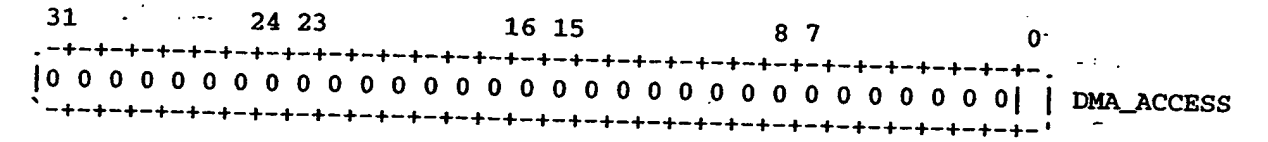
This register should only be accessed when PGRAPH:STATUS:STATE is Idle regardless or whether it is a erad or a write access. This register does not need to be context switched. It is provided for diagnostic accesses only.




```
#define NV_PGRAPH_CACHE_RAM                    0x004006c4 /* RW-4R */
#define NV_PGRAPH_CACHE_RAM_VALUE                    31:0 /* RWXVF */
```

Figure 9-19 Cache Data Access register

The ACCESS register enables the DMA engine after a DMA_INTR_0:PRESENT interrupt. When a DMA_INTR_0:PRESENT interrupt is detected, the DMA_TLB_PTE register should be loaded with the appropriate information. If DMA_CONTROL_PAGE_TABLE:ENABLE, then the instance PTE may also be updated. The ACCESS_PTE:ENABLE signals the DMA engine to continue the DMA transaction. *Note. DMA_INTR_0:INSTANCE, DMA_INTR_0:PROTECTION, DMA_INTR_0:LINEAR are fatal interrupts. The DEBUG_1_DMA_ACTIVITY:CANCEL must be asserted to cancel the outstanding dma request.



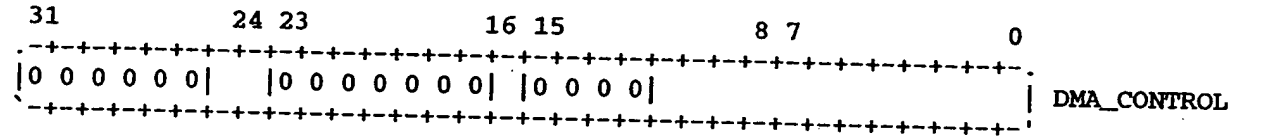

```
#define NV_PGRAPH_DMA_ACCESS                                  0x00401200 /* -W-4R */

#define NV_PGRAPH_DMA_PTE_ACCESS_PTE                             0:0 /* -W-VF */
#define NV_PGRAPH_DMA_PTE_ACCESS_PTE_ENABLE                0x00000001 /* -W--V */
```

Figure 13-1 Enable Register

The CONTROL registers contain miscellaneous information about the current DMA instance.

The ADJUST bits contain the adjustment from the physical page boundary to the user specified DMA base address. This is normally the low 12 bits of the DMA base address (plus the Selector base for the 80x86). The ADJUST is added to the offset to make it page boundary aligned before storing the result in PGRAPH_DMA:ADJ_OFFSET.

The PAGE_TABLE bit indicates that the page table is present in the memory immediately following the DMA Instance in PRAMIN. PAGE_TABLE is set to NotPresent when there is not enough memory in PRAMIN to hold the page table.




```
#define NV_PGRAPH_DMA_CONTROL                                 0x00401210 /* RW-4R */

#define NV_PGRAPH_DMA_CONTROL_ADJUST                             11:0 /* RWXUF */
#define NV_PGRAPH_DMA_CONTROL_PAGE_TABLE                         16:16 /* RWXVF */
#define NV_PGRAPH_DMA_CONTROL_PAGE_TABLE_NOT_PRESENT       0x00000000 /* RW--V */
#define NV_PGRAPH_DMA_CONTROL_PAGE_TABLE_PRESENT           0x00000001 /* RW--V */
#define NV_PGRAPH_DMA_CONTROL_TARGET_NODE                        25:24 /* RWXUF */
#define NV_PGRAPH_DMA_CONTROL_TARGET_NODE_NVM              0x00000000 /* RW--V */
#define NV_PGRAPH_DMA_CONTROL_TARGET_NODE_CART             0x00000001 /* RW--V */
#define NV_PGRAPH_DMA_CONTROL_TARGET_NODE_PCI              0x00000002 /* RW--V */
#define NV_PGRAPH_DMA_CONTROL_TARGET_NODE_AGP              0x00000003 /* RW--V */
```

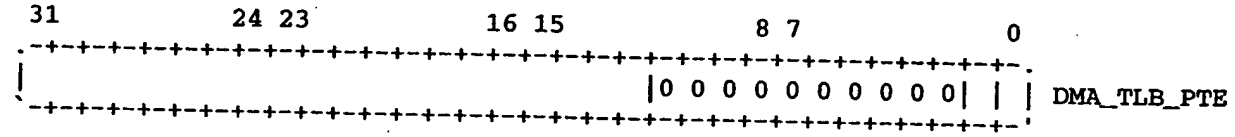
Figure 13-2 Control Registers

The TLB_PTE registers contain the virtual to physical address translation lookaside buffer's page table entry.

The PAGE bit indicates that the page is present in memory. This is a logic AND of a set of conditions maintained by the Resource Manager such as the page is present in memory, locked down, user permission, etc. No access is made to main memory unless PAGE is set to Present. When the DMA engine loads a new control/limit/PTE from PRAMIN and the PGRAPH_DMA:CONTROL:PAGE_TABLE is set to NotPresent, the DMA engine will set PAGE to NotPresent instead of loading the PTE. An optional interrupt can be generated on access to not present pages by setting PGRAPH_DMA:INTR_EN_0:PRESENT.

The ACCESS bit indicates if the page is writable. All pages are readable. Any writes to a read only page in main memory be masked. An optional interrupt can be generated on illegal writes by setting PGRAPH_DMA: INTR_EN_0:PROTECTION.

The FRAME_ADDRESS bits contain the page frame address. These are the upper 20 bits of the physical page frame of the page in the DMA TLB. The tag for the TLB_PTE is in PGRAPH_DMA:TLB_TAG. If the tag matches the upper 20 bits of the PGRAPH_DMA:ADJ_OFFSET, the low 12 bits of the offset is combined with the
upper 20 bits of the FRAME_ADDRESS to create the physical address. If the tag does not match, the correct TLB_PTE is read from the DMA Instance page table (if present). The resource manager must ensure that none of the Page Frame Address values in any of the TLB_PTE registers refer to any of the physical addresses assigned to the NV chip. The chip cannot DMA to or from its own address space.

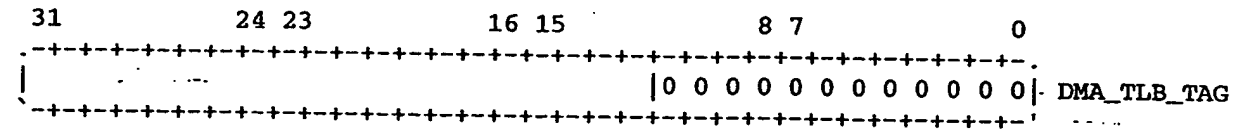



```
#define NV_PGRAPH_DMA_TLB_PTE                                  0x00401230 /* RW-4R */

#define NV_PGRAPH_DMA_TLB_PTE_PAGE                                    0:0 /* RWXVF */
#define NV_PGRAPH_DMA_TLB_PTE_PAGE_NOT_PRESENT                 0x00000000 /* RW--V */
#define NV_PGRAPH_DMA_TLB_PTE_PAGE_PRESENT                     0x00000001 /* RW--V */
#define NV_PGRAPH_DMA_TLB_PTE_ACCESS                                  1:1 /* RWXVF */
#define NV_PGRAPH_DMA_TLB_PTE_ACCESS_READ_ONLY                 0x00000000 /* RW--V */
#define NV_PGRAPH_DMA_TLB_PTE_ACCESS_READ_WRITE                0x00000001 /* RW--V */
#define NV_PGRAPH_DMA_TLB_PTE_FRAME_ADDRESS                         31:12 /* RWXUF */
```

Figure 13-4 TLB Page Table Entry Registers

The TLB_TAG registers contain the virtual to physical address translation lookaside buffer's tag for the page table entry currently in PGRAPH_DMA:TLB_PTE.

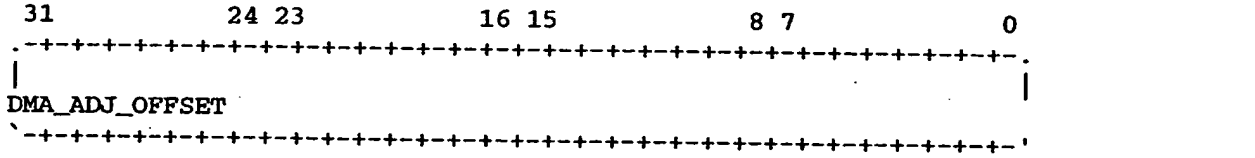
The ADDRESS bits contain the upper 20 bits of the adjusted offset address for the current TLB's page table entry. ADDRESS is compared to the new adjusted offset. If they match, then the PGRAPH_DMA:TLB_PTE:FRAME_ADDRESS is loaded with the correct information.




```
#define NV_PGRAPH_DMA_TLB_TAG                                   0x00401240 /* RW-4R */

#define NV_PGRAPH_DMA_TLB_TAG_ADDRESS                             31:12 /* RWXUF */
```

Figure 13-5 TLB Tag Registers

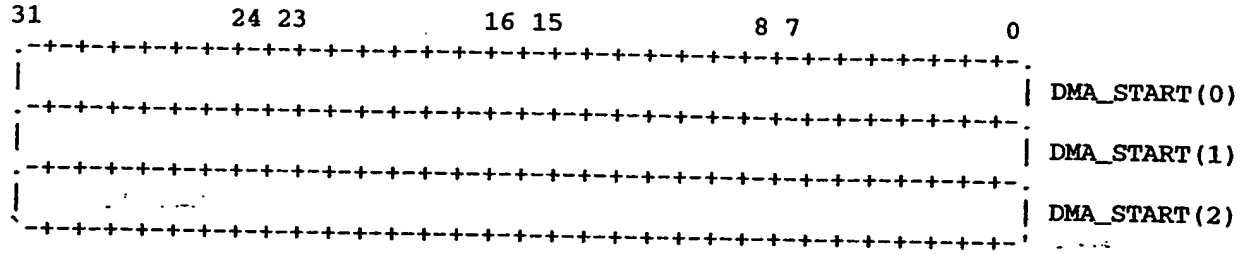
The ADJ_OFFSET registers contain the adjusted (page aligned) offset from the user DMA base address. ADJ_OFFSET contains the sum of PGRAPH_DMA:OFFSET and PGRAPH_DMA:CONTROL:ADJUST. The upper 20 bits of the ADJ_OFFSET register is used to lookup the page table entry and to compare to PGRAPH_DMA:TLB_TAG. The PGRAPH_DMA:DMA_TLB_PTE_PAGE address and the low 12 bits of ADJ_OFFSET are used to form the physical address for the DMA.




```
#define NV_PGRAPH_DMA_ADJ_OFFSET                                0x00401250 /* RW-4R */

#define NV_PGRAPH_DMA_ADJ_OFFSET_VALUE                            31:0 /* RWXUF */
```

Figure 13-6 Adjusted Offset Register

The DMA_START[] value holds the START method value for DMA classes. This value is used to create the offset value used by the DMA engine to calculate the virtual address of the intended access. This register should only be written when PGRAPH:STATUS:STATE is Idle. It can be read at any time.




```
#define NV_PGRAPH_DMA_START(i)                    (0x00401800+(i)*16) /* RW-4A */
#define NV_PGRAPH_DMA_START__SIZE_1                                 3 /*       */

#define NV_PGRAPH_DMA_START_VALUE                                  31:0 /* RWXUF */
```

Figure 13-11 Dma Starting Offset register